Scheduling trains with delayed departures

Fernand Meyer

Scheduling trains with delayed departures

Fernand Meyer

Mines ParisTech,
Département maths et systèmes,
Centre de Morphologie Mathématique,
F-77305 Fontainebleau Cedex, France

Abstract. A number of image transformations may be expressed as shortes paths algorithms in various algebras We introduce a general framework where a graph is considered as a railway network on which the trains follow shortest paths. Given a fixed traveling time on each edge and an arbitrary distribution of departure times, one searches the first arrival time of a train at each node. This formulation leads to interesting filters and image decomposition in the $(\min, +)$ algebra and sheds new light on flooding and razing algorithms in the algebras $(\min, \max)$ and $(\max, \min)$.

1 Introduction

Reconstruction openings and closings, levelings and flattenings [3], all these operators transform an input image under constraints coming from another image. For instance the reconstruction opening of an image f with a marker function g is the highest flooding of f under g. We reinterpret these operators in terms of shortest paths algorithms. The highest flooding of f under g may be seen as the schedules of trains circulating on a graph, at departure times derived from g and traveling times derived from f. These shortest paths are expressed in a particular path algebra, the $(\min, \max)$ algebra.

Gondran and Minoux [1] have shown that many problems in graph theory amount to searching shortest paths in a particular algebra. The common structure of these algebra is a dioïd ; they differ by the laws by which the lengths of the paths on a graph are measured and compared. For instance the algebra (min, +) deals with the ordinary shortest paths: the second operator, here +, permits to define the length of a path as the sum of the lengths of its edges ; the first operator permits to compare two paths, the shortest path being the one with the shortest length. In the algebra $(\min, \max)$, the "length" of a path is the greatest weight of its edges ; the "shortest" will be the path for which this highest weight is the smallest. This algebra is familiar in morphological segmentation, as any flooding always follows a shortest path in this algebra. In a dual way, in the algebra $(\max, \min)$ the "length" of a path is the smallest weight of its edges ; the "shortest" will be the path for which this smallest weight is highest.

The triangular inequality immediately proceeds from the two operators of the algebra. It relies on the fact that the shortest path between two nodes x and

z is shorter than the length of the path obtained by concatenating the shortest path between x and a node y on one hand and the shortest path between y and z on the other hand.

In the algebra $(\min, +)$ one gets the classical triangular inequality $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$. In the algebra $(\min, \max)$ one gets the ultrametric inequality $\delta(x, z) \leq \delta(x, y) \vee \delta(y, z)$, stronger than the triangular inequality. In the algebra $(\max, \min)$ one gets the dual ultrametric inequality $\delta(x, z) \geq \delta(x, y) \wedge \delta(y, z)$.

We introduce a general framework where a graph is considered as a railway network on which the trains follow the shortest paths. Given a fixed traveling time on each edge, and an arbitrary distribution of departure times, one searches the first arrival time of a train at each node. This formulation leads to interesting filters and image decompositions in the $(\min, +)$ algebra and sheds new light on flooding and razing algorithms in the algebras $(\min, \max)$ and $(\max, \min)$.

2 Scheduling trains in the algebra $(\min, +)$

2.1 Setting the scenario of trains circulating on graphs

Let $G = (N, E)$ be a graph where nodes N and oriented edges or arrows E are weighted with positive weights. The graph may be considered as a railway network, where the nodes are railway stations and the arrows are connections between them. Trains may follow all possible paths on G. Consider a particular train. It starts at station $s \in N$ at time $\tau(s)$, follows a path $\theta = (x_0 = s, x_1, ..., x_n = t)$ where x_i and x_{i+1} are two railway stations linked by an arrow $(i, i+1)$ of E, weighted by the time $e_{i,i+1}$ needed for following it. The arrival time at destination is then $\tau(s) + \sum_{i,i+1 \in \theta} e_{i,i+1}$.

We now consider all trains starting at all possible nodes and taking all possible routes, and observe the earliest time when each node is reached by a train; if a train arrives at i before $\tau(i)$, we replace $\tau(i)$ by this first arrival time. For some nodes no train ever arrives ; i is such a node, if $\tau(i)$ is too early, so that all trains coming from another node arrive at i after $\tau(i)$. For others, no train departs: if $\tau(i)$ is too late, no train starting from i has a chance to be the first to reach another node ; this is in particular the case if $\tau(i) = \infty$. Some nodes cumulate both situations, and no train departs or arrives.

The resulting shedules $\widehat{\tau} = \Theta(e, \tau)$ depend on the distribution of initial departure times and travelling times along each edge. It is an opening on τ, as it is obviously anti-extensive and increasing. It is also idempotent, as a second scheduling would not change the distribution $\tau(i)$ any further.

Scheduling the trains highlights a number of structures in the graph, associated to the distribution of weights on nodes and edges:

- time flat zones: maximal connected components for which it takes no time for travelling from a node to the next.
- nodes or time flat zones where trains only start.
- nodes or time flat zones where trains only arrive.

– nodes or time flat zones where no train appears.
– maximal routes, which are geodesics of the traveling times.
– at any node, measure of the number of trains passing through this node.

Let us now present how to easily schedule the trains, given a distribution of traveling times e on edges and an initial distribution of departure times τ.

Remark 1. For a node distribution where all nodes get the initial weight ∞ except the node s for which $\tau(s) = 0$, the scheduling produces the shortest path between s and all others. The same holds if we replace s by all nodes of a collection X of nodes.

2.2 Scheduling the trains

Shortest path in an augmented graph. We now fix G and e and consider varying distributions of starting times. Any such distribution is a function defined on the nodes of G. In case of images defined on a grid, G is the graph associated to the grid: pixels are nodes and neighboring pixels are connected by an edge.

We obtain an augmented graph G_Ω, by adding to G a dummy node Ω with weight $\tau(\Omega) = 0$ and dummy edges (Ω, i) between Ω and each node i of N, with $e_{\Omega i} = \tau_i$. Since the travelling time along the edge (Ω, i) is τ_i, it is equivalent for a train to start from i at time τ_i or from Ω at time 0 and follow the edge (Ω, i) for reaching i. The earliest time for a train to arrive at any node k of G_Ω is the total duration of the shortest path between Ω and k. It follows that scheduling the graph G amounts to constructing the shortest paths between Ω and all other nodes in the graph G_Ω, which is a classical problem in graph theory for which many algorithms exist. Here we consider the algorithms of Moore Dijkstra and of Berge [1]. The first is a fast and greedy algorithm. The second may be implemented for images with local operators requiring a simple raster scan, very easy to realize in hardware.

Algorithm of Moore Dijkstra The algorithm takes $|N|$ steps. At any step, S represents the subset of nodes for which the shortest path is known. For any neighboring node of S, the length of the shortest path for which all edges but the last belong to S constitutes an overestimation of this length. However the node for which this overestimation is the lowest is correctly estimated. The algorithm proceeds by incorporating this node into S and updating its neighborhood.

Initialisation:
$S = \Omega$ and $\overline{S} = N$
While $\overline{S} \neq \varnothing$ repeat:
Select $j \in \overline{S}$ for which $\tau(j) = \min_{i \in \overline{S}} [\tau(i)]$
$\overline{S} = \overline{S} \backslash \{j\}$
For any neighbor i of j in $\overline{S}$ do $\tau(i) = \min [\tau(i), \tau(j) + e_{ji}]$
End While

Fig. 1. Synchronisation $\Theta(f_{ij}^4, f)$ of image f in the graph G_f^4

Remark 2. Introducing the dummy node Ω and the dummy edges helps us in reformulating the scheduling problem in a known framework. It is however useless in practice and the algorithm gives the same results by replacing during initialisation the instruction $S = \Omega$ by $S = \varnothing$.

Algorithm of Berge The shortest path to j may come from its neighbor i and then $\tau(j) = \tau(i) + e_{ij}$. If it is not the case we have $\tau(j) \leq \tau(i) + e_{ij}$. The algorithm scans the image and corrects the values of τ until the previous inequality is satisfied everywhere.

As long there exists an arrow (i, j) such that $\tau(j) > \tau(i) + e_{ij}$ do $\tau(j) = \tau(i) + e_{ij}$

If one considers all neighbors of j at the same time the algorithm becomes $\tau(j) = \tau(j) \wedge \bigwedge_{i \text{ neighbor of } j} (\tau(i) + e_{ij},)$

Remark 3. The algorithm makes no use of Ω nor of the dummy edges, since the relation $\tau(j) \leq \tau(\Omega) + e_{\Omega j}$ is satisfied by construction: $\tau(\Omega) + e_{\Omega j} = 0 + \tau(j)$.

2.3 Illustration

In case of images defined on a grid, the nodes N of the graph G are the pixels and the arcs E are the edges between neighboring pixels. In our illustration we take an image f represented left in figure 1. The valuation of the arrow from i to j is equal to $f_{ij}^4 = \vee(f_j - f_i, 4)$; only the ascending transitions with an amplitude higher than 3 are considered, in all other cases the weight is 0. As these traveling times are much smaller than the pixel values represented on the nodes, the scheduling $\Theta(f_{ij}^4, f)$ represented right in figure 1 has a dramatic effect of filtering and simplification of the initial image. It highlights all sharp transitions and completely discards the noise dark or bright noise particles with less contrast.

In "Image decompositions and transformations as peaks and wells" in the same issue, we show how to decomposes an image as a difference between a peaks component and a wells component. Both components are obtained by scheduling two distinct graphs. Both have as node weights the valuations of the pixels and as edge weights respectively the upwards and downwards transitions in the image.

3 Scheduling trains in the algebra (min, max)

We consider the same framework as previously, same graph, same weight interpretation, same circulation rules of trains. Durations of the journey and arriving times of the trains are however computed differently. In the algebra (min, max), the arrival time of a train starting at s and following a path θ is the maximum between the starting time and the traversal time of all followed edges $(i,j) \in \theta$; it is equal to $\tau(s) \vee \bigvee_{i,j \in \theta} e_{ij}$. In this algebra, the arrival time of a train is equal to its departure time as soon as this time is higher than the crossing time of the edges along the path. For such a path, the passing time of the train at each railway station is the same. Such a trajectory belongs to "a time flat zone".

The shortest path algorithms are still valid if one replaces the operator + by the operator max. The scheduling of trains, given a fixed distribution e of weights on the edges and an initial distribution of weights τ on the nodes will be written $\overline{\Theta}(e,\tau)$. Adding the same dummy node and edges as previously permits to reformulate the scheduling problem as a shortest path problem between Ω and all other edges. The algorithms of Moore-Dijkstra and Berge remain the same if one replaces the operator + by the operator $\vee$.

It is possible to reformulate Berge's algorithm with the help of two adjunctions between edges and nodes on a graph
- an erosion from the nodes to the edges $[\varepsilon_{en} n]_{ij} = n_i \wedge n_j$ with its adjunct dilation $[\delta_{ne} e]_i = \bigvee_{(k \text{ neighbors of } i)} e_{ik}$
- a dilation from the nodes to the edges $[\delta_{en} n]_{ij} = n_i \vee n_j$ with its adjunction erosion $[\varepsilon_{ne} e]_i = \bigwedge_{(k \text{ neighbors of } i)} e_{ik}$

For storing intermediate results we will need a second set of valuations l on the edges of the graph. If the shortest path follows an edge (i,j), it leaves the smallest node at time $\tau(i) \wedge \tau(j) = (\varepsilon_{en}\tau)_{ij}$ and arrives at the largest at time $l_{ij} = (\varepsilon_{en}\tau)_{ij} + e_{ij}$. Now, the final schedule attached to i will be $\tau(i)$ if no path arrives earlier ; in the other case, it reaches i at the earliest time of any path passing through one of the adjacent paths, that is $\bigwedge_{j \text{ voisin de } i} l_{ji} = (\varepsilon_{ne} l)_i$. Thus the shortest path at i has the value $(\tau \wedge \varepsilon_{ne} l)_i$. Finally, if the preceding relations will be satisfied if we repeat until stability $\{l = \varepsilon_{en}\tau + e \ ; \ \tau = \tau \wedge \varepsilon_{ne} l\}$. In [4] we derived this algorithm from considerations on the relationt between flooding levels and heights of pass points on a graph.

Minimum spanning tree The shortest distances in the algebra (min, max) are called ultrametric distances, as they verify the ultrametric inequality presented above. The theorem of Hu (1961) states that all shortest paths belong to a minimum spanning tree (MST) of the graph. This implies that all problems implying shortest path may be solved using an MST of the graph (the MST is unique if all edges have different weights). Like that, one has to process far less neighbors. These distances are closely linked to flooding. The trains starting at s and arriving at p follows the same route as a flood where the source would

be placed at s. The result of any scheduling in the $(\min, \max)$ algebra also is a scheduling.

3.1 Scheduling and floodings on graphs

A criterion for recognizing a flooding on graphs. We now establish a local criterion which characterizes the schedules of the trains on the graph, or equivalently the floodings on this graph.

Consider two neighboring pixels p and q on a geodesic path, they verify $\tau(q) = \tau(p) \vee e_{pq}$. The flood going from p to q remains at the same level if $e_{pq} < \tau(q)$, and climbs at level e_{pq} in the opposite case. If (p, q) do not belong to a geodesic path, we only have an inequality $\tau(q) \leq \tau(p) \vee e_{pq}$. The following expressions are all equivalent and characterize floodings on graphs: $\tau(q) \leq \tau(p) \vee e_{pq} \Leftrightarrow [\tau(q) \leq \tau(p)$ or $\tau(q) \leq e_{pq}] \Leftrightarrow [not\{\tau(q) > \tau(p)\}$ or $\tau(q) \leq e_{pq}] \Leftrightarrow \{\tau(q) > \tau(p) \Rightarrow \tau(q) \leq e_{pq}\}$

The interpretation of this last implication is obvious: the flooding level can stay at level $\tau(q) > \tau(p)$ only if the edge e_{pq} verifies $\tau(q) \leq e_{pq}$ and prevents its leakage towards the lower level of $\tau(p)$.

This criteria are useful for recognizing whether a distribution of weights on nodes ais a flooding on an edge weighted graph. It permits to show that the maximum and minimum of two floodings still is a flooding. Let μ and ν be two floodings of G. They both verify the first criterion, and so do $\mu \vee \nu$ and $\mu \wedge \nu$ showing that the also are floodings of G.

If we now consider an arbitrary distribution μ of weights on G, the supremum of all floodings of G below μ is nothing but the scheduling $\overline{\Theta}(e, \mu)$. Indeed, if we follow a geodesic path downwards, the lowest node verifies $\overline{\Theta}(e, \mu) = \mu$, which shows that this flooding path could not be higher without offending the condition $\overline{\Theta}(e, \mu) \leq \mu$.

3.2 Morphological applications

Scheduling the neighborhood graph

Hierarchical segmentation We start from a partition of a domain D, for which a dissimilarity between neighboring regions has been defined. For segmenting an image, one often takes the catchment basins of is gradient image as partition. Each region being represented as a node of a graph ; adjacent regions being linked by an edge weighted by the height of the pass point between. Cutting all edges with a weight above a threshold produces a forest, where each tree represents a region of a partition which is coarser than the original one. For increasing values of λ, the partitions become coarser and coarser.

The same coarse partition may be obtained by flooding. Assigning a constant value $\tau(i) = \lambda$ to all nodes constitutes a valid flooding of any edge weighted graph, as the criterion $\tau(q) \leq \tau(p) \vee e_{pq}$ is everywhere satisfied. Cutting all edges with a weight $e_{ij} > \lambda$, is the same as cutting the edges verifying $e_{ij} > \tau(i) \vee \tau(j)$ for this uniform distribution of weights.

The mechanism is however general, and one may use any distribution of weights τ for generating coarser partitions. The scheduling $\widehat{\tau} = \overline{\Theta}(e, \tau)$ is a flooding of the graph which may be submitted to the same pruning: cutting all edges verifying $e_{ij} > \widehat{\tau}(i) \vee \widehat{\tau}(j)$ produces a coarser graph. The distribution τ may be tailored in order to favour the segmentation of particular structures. For instance, swamping is a particular case where τ is equal to ∞ except at the position of markers.

Flooding images through their neighborhood graph A flooded image is entirely known if one knows for each of its catchment basins whether it contains or not a lake and if it is the case, the level of this lake. This remark is at the basis of the following algorithm for obtaining the highest flooding of a function f below a function g.

First one constructs the region adjacency graph of f, each node i representing a catchment basin R_i ; an edge links neighboring basins R_i and R_j with a weight equal to th height of the pass point between them. This weight distribution permits to model the flood progression, as it crosses the path points from one basin to the next. The weight $\tau(i)$ of node i is the minimum altitude of g inside the catchment basin R_i. The scheduling $\overline{\Theta}(e, \tau)$ produces the height of the flood in each region R_i. It is of course much faster to compute the flooding on the graph or its MST than on the initial image. However, this graph and MST has first to be constructed. We have shown in [4] that it becomes interesting if one has a few levelings to do on the same image.

Filtering Scheduling may also be used for filtering and simplifying the image represented on the nodes. We schedule two graphs derived from the image 2. The nodes are initialized with the function τ_0, equal to 0 on the upper side of the image and to ∞ everywhere else. Both graphs have different valuations on the edges.

- in G_f^+, the arrow e_{ij} from i to j gets the valuation $f_{ij}^+ = \vee(f_j - f_i, 0)$; the value is positive only for increasing transitions
- in G_f^-, the arrow e_{ij} from i to j gets the valuation $f_{ij}^+ = \vee(f_i - f_j, 0)$; the value is positive only for decreasing transitions

The schedules $\overline{\Theta}(f_{ij}^+, \tau_0)$ and $\overline{\Theta}(f_{ij}^-, \tau_0)$ are presented in positions 2.1 and 3.1 of figure 2. Bright noise particles appear in the positive transitions and dark noise particles in the negative transitions. Filtering these noise components thanks to a reconstruction opening associated to size and contrast produces the images 2.2 and 2.3. Subtracting the minimum of these images from the image 2.3 produces a clean image, represented in position 1.2.

Equivalence between flooding on graphs and flooding on images

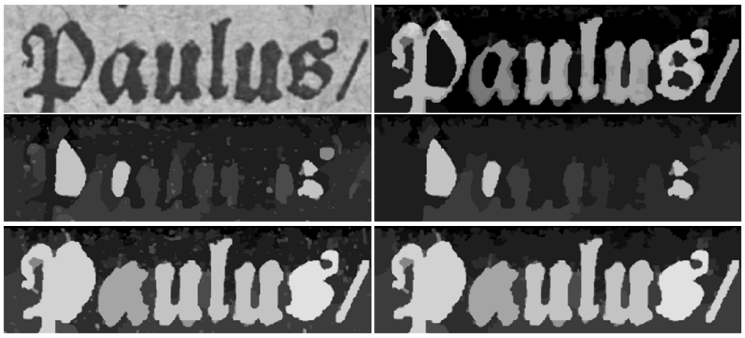

Fig. 2. Harmonization in a positive and a negative part, filtering and reconstruction of the filtered result.

Reformulation of flooding of images as a flooding of graphs We now have two related notions: floodings as reconstruction closings on images and floodings as schedulings on graphs in the algebra $(\min, \max)$. The first operates on images at the pixel level, takes two images f and g as arguments, and produces the highest flooding of f under g. The second operates on an edge and node weighted graph, starts from an initial distribution of weights τ on nodes, and produces by scheduling $\overline{\Theta}(e, \tau)$ a distribution of weights on the nodes which may be interpreted as a flooding. We now establish the correspondence between these two types of flooding.

Floodings on images are characterized by a local criterion *imageflood* [3]: $h \geq f$ is a flooding of f if for any couple of neighboring pixels (p, q), we have: $h_p > h_q \Rightarrow f_p \geq h_p$.

As established above, floodings on graphs are characterized by the criterion *graphflood*: $\tau(q) > \tau(p) \Rightarrow \tau(q) \leq e_{pq}$ where p and q are neighboring nodes.

Images are defined by their values on the pixels of a grid. As noted above, this grid may be considered as a graph. The nodes are the pixels, the edges connect neighboring nodes. We call it grid graph. Consider two functions f and g verifying $g \geq f$. The image flooding algorithm: $g^{(0)} = g$ and $g^{(n+1)} = f \vee \varepsilon g^{(n)}$ produces a decreasing sequence converging towards the flooding $g^{(\infty)}$.

We now show, that with appropriate weights on the nodes and edges of the grid graph, the same result may be obtained by graph flooding. On the grid graph, we assign to the nodes the valuation $\tau(i) = g(i)$ and to the edges $e_{ij} = f_i \vee f_j$. The algorithm of Berge progressively decreases g: every time $\tau(i) > \tau(j) \vee e_{ij}$, $\tau(i)$ is replaced by $\tau(j) \vee e_{ij}$. Replacing τ and e by their values shows that we replace g_i by $g_j = g_j \vee f_i \vee f_j$, which garanties that all along the process g verifies $g \geq f$. The flooding obtained at convergence verifies the criterion *graphflood* which becomes here: $g_p > g_q \Rightarrow g_p \leq e_{pq} = f_p \vee f_q$. Since $g_p > g_q \geq f_q$, the relation $g_p \leq f_p \vee f_q$ simply becomes $g_p \leq f_p$ which precisely is the criterion *imageflood* indicating that g is a flooding of f, now considered as images.

Algorithmic consequences For flooding an image f, it is not necessary to use all edges of the images but only the edges which belong to the minimum spanning tree of the graph weighted by $f_i \vee f_j$.

The algorithm of Moore Dijkstra may be implemented using a hierarchical queue.

Initialisation: each pixel i of the image is introduced into a hierarchical queue HQ with priority g_i

While HQ is not empty:

- extract i, the node with the highest priority

- label i as treated

- introduce in HQ all untreated neighbors j of i, with the prioriry $g_j = g_j \wedge (f_j \vee \min_{i \in S(j)} g_i) = f_j \vee \left[g_j \wedge \min_{i \in S(j)} g_i)\right]$, where $S(j)$ are the neighbors of j which are already treated.

End while

The Dijkstra algorithm is a greedy algorithm, i.e. it processes each pixel only once and produces the correct response. Furthermore, it requires checking only the neighbors of j belonging to S, and not the complete neighborhood of j, as do the usual algorithms. Finally, this neighborhood may be further reduced to the neighbors of j on the MST if the tree is already known.

The minimum spanning tree of steepest descent and the watershed As a conclusion of the preceding section, we proved that scheduling the grid graph with the weights $\tau(i) = g(i)$ for the nodes and $f_i \vee f_j$ for the edges produces the same result as flooding the function f with the marker g. As the same weight $f_i \vee f_j$ is given to each edge linking the node i to each of its lower neighbors, there are many possibilities to construct distinct minimum spanning trees, which may be indifferently followed by the flow, yielding the same end result.

Now, constructing the watershed line may also be obtained as a flooding: it starts with the regional minima and follows the lines of steepest descent. This will be the case if each pixel is flooded through its lowest neighbor. Assigning a distinct label to each minimum and propagating the labels all along the flooding permits to obtain a partition of the image where each region is a catchment basin with the same label as the minimum at the source of its flooding.

Consider again the grid graph of f with its edge valuations $f_i \vee f_j$. Among all its MSt, there is one which contains the lines of steepest descent for f: each node is connected with one of its lowest neighbors. The classical algorithm [2] for constructing the watershed propagates the labels of the minima along this tree T.

Initialisation: assign a distinct label to each minimum and put the boundary pixels of the minima in a hierarchical queue HQ.

While HQ is not empty, extract p, the node with the highest priority:

- for each neighbor q of p without label, do $label(q) = label(p)$, introduce q in HQ, and add the edge (p, q) to T with the weight $f_q = f_p \vee f_q$ (p being the first neighbor to exit the HQ, is indeed one of the lowest neighbors of q)

– if q is neighbor of p with a label and if both labels meet for the first time, add the edge (p, q) to T with an edge valuation $f_p \vee f_q$ (this edge corresponds to a pass point between two catchment basins.

End while

The tree T is indeed a minimum spanning tree, and the valuation of its edges (p, q) are $f_p \vee f_q$. If we have a series of markers, we assign a distinct label to each marker, and produce a function τ equal to f on the markers and to ∞ everywhere else. The scheduling $\overline{\Theta}(e, \tau)$ on T with label propagation permits to obtain the watershed associated to the markers. The catchment basins where a marker is present are flooded from this marker, whereas the catchment basins without marker are flooded from neighboring basins. Paying the cost of constructing T becomes interesting only if several distinct segmentations have to be obtained for the same image.

4 Scheduling in the algebra (max, min)

The algebra $(\max, \min)$ is dual to the algebra$(\min, \max)$. Hence all results are updated if one replaces in the preceding section:

– max by min and min by max
– $\wedge$ by $\vee$ and $\vee$ by $\wedge$
– minimum spanning tree by maximum spanning tre

Floodings become razings.

5 Conclusion

Considering shortest paths with delayed departure times permits to unify a number of algorithms under a common algorithmic framework. One may then chose the algorithm which is best suited to a given framework, or best adapted to the primitives which are present in a particular library or in a particular hardware. The scheduling algorithm enlarges to the $(\min, +)$ algebra the rich world of transformations, where an image is transformed under constraints derived from another image, as is the case in the $(\min, \max)$ algebra with reconstruction closings and in the $(\max, \min)$ algebra with reconstruction openings.

References

1. M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1995.
2. F. Meyer. Topographic distance and watershed lines. *Signal Processing*, pages 113–125, 1994.
3. F. Meyer. The levelings. In H. Heijmans and J. Roerdink, editors, *Mathematical Morphology and Its Applications to Image Processing*, pages 199–207. Kluwer, 1998.
4. F. Meyer and R. Lerallut. Morphological operators for flooding, leveling and filtering images using graphs. In F. Escolano and M. Vento, editors, *Graph-Based Representations in Pattern Recognition*, volume 4538 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin / Heidelberg, 2007.